****

**Bilkent University**

# **Introduction to Cyber Security**

**Take Home exam**

**Mert Özerdem**

**21300835**

**Victim(HostS)’s IP: 10.10.21.85 (ubuntu)**

**Attacker(HostC)’s IP: 10.10.21.30 (ubuntu)**

**About Environment**: Both machines run (linux) ubuntu 14.04.

**Port Scanning:** Port scanning is the process of attempting to connect to a number of sequential ports, for the purpose of acquiring information about which are open and what services and operating system are behind them.[1]

Ports are specified by a number ranging from 1 to 65535.

* Many ports below 1024 are associated with services that Linux and Unix-like operating systems consider critical to essential network functions, so you must have root privileges to assign services to them.
* Ports between 1024 and 49151 are considered "registered". This means that they can be "reserved" (in a very loose sense of the word) for certain services by issuing a request to the IANA (Internet Assigned Numbers Authority). They are not strictly enforced, but they can give a clue as to the possible services running on a certain port.
* Ports between 49152 and 65535 cannot be registered and are suggested for private use.[2]

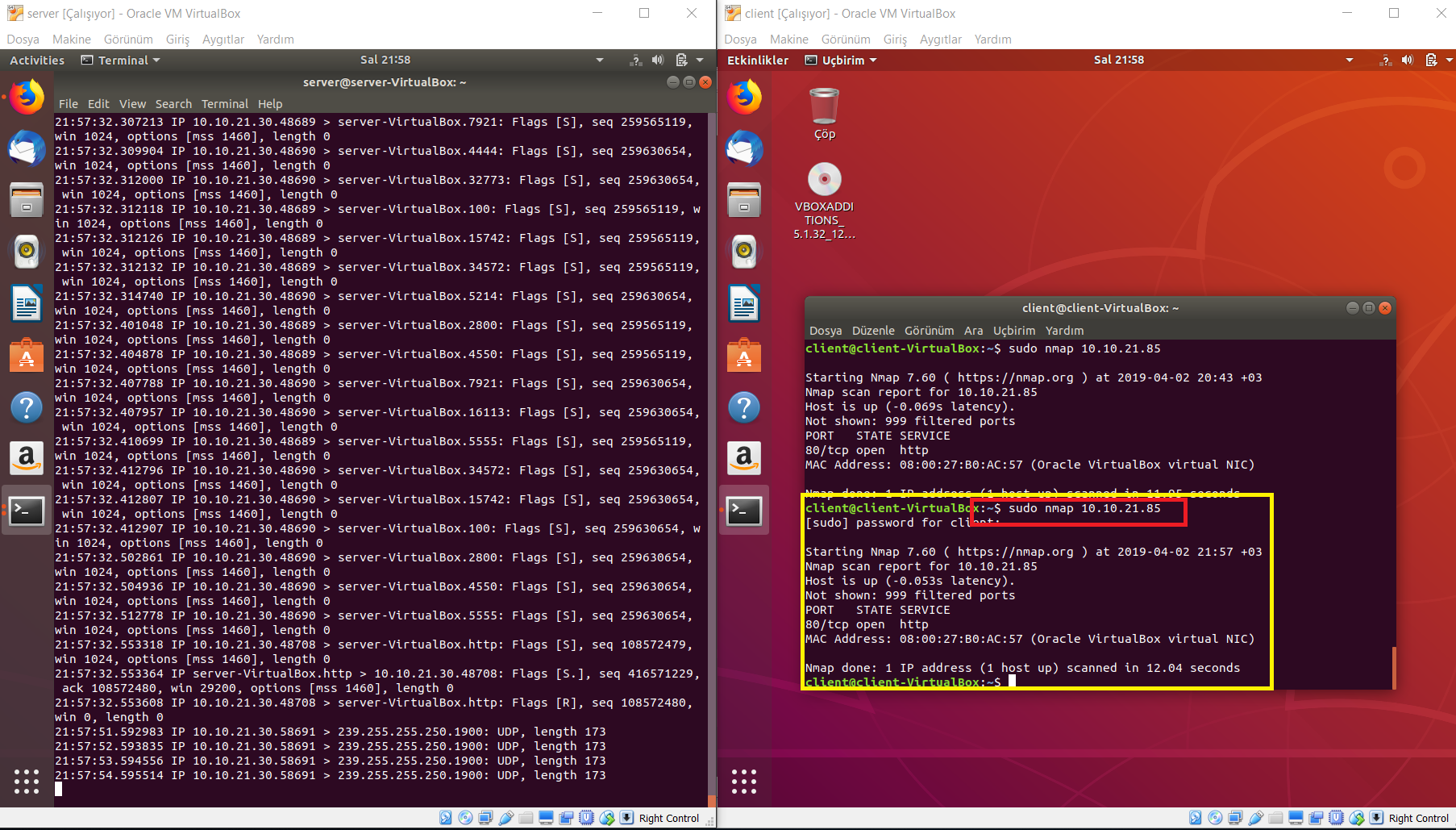
This is the definition of the ports and how they are used by OS.

First, we make an ping sweep attack and get the information about which IPs are active. This is not included in this report due to the restrictions that are made in the Take Home exam paper.

**T1)Port Scan**

In this part of the project, HostC starts an port scan on the HostS. HostC uses the nmap command;

**sudo nmap 10.10.21.85**

as we can see in the below picture.

From the above screenshot, we can see that HostC starts an port scan with to scan the IP 10.10.21.85 which is the HostS(Server). Purpose of this is to acquiring information about which ports are open and what services and operating system are behind them. Red circled part is the nmap command and yellow circle is the output of the command at the red circled part.

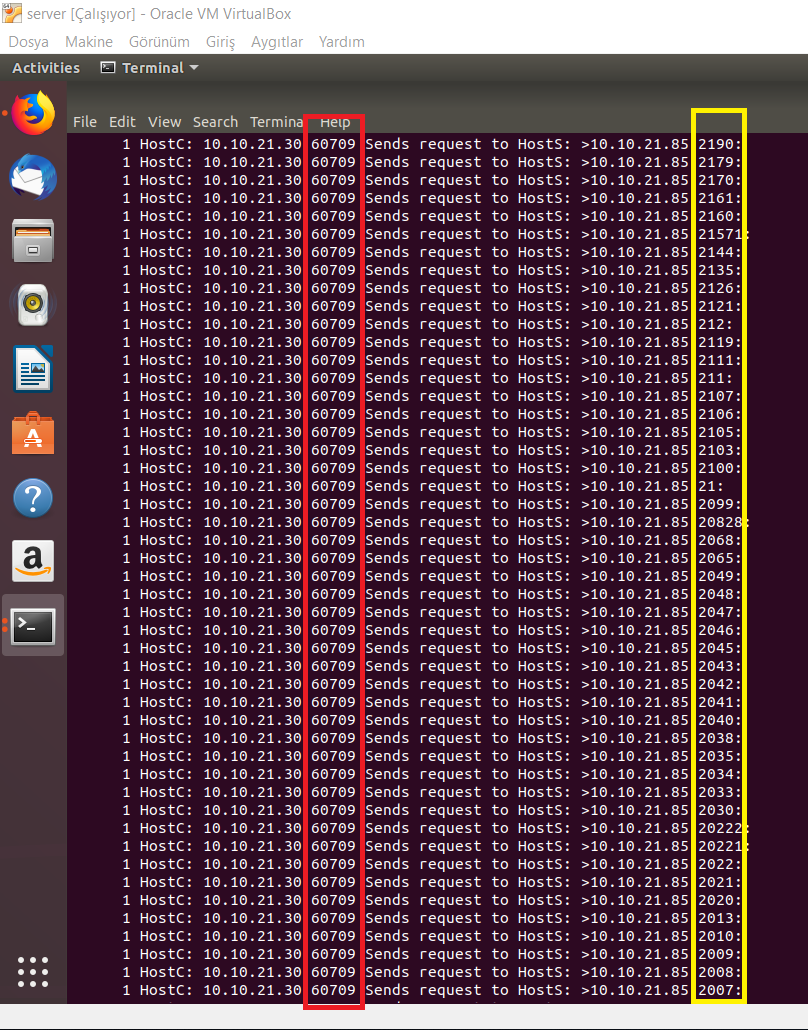
**T2)Analysis of Packet Capture**

Normally to start a TCP connection, the requesting end sends a "synchronize request" packet to the server. The server then sends a "synchronize acknowledgment" packet back. The original sender then sends back an "acknowledgment" packet back to the server, and a connection is established.

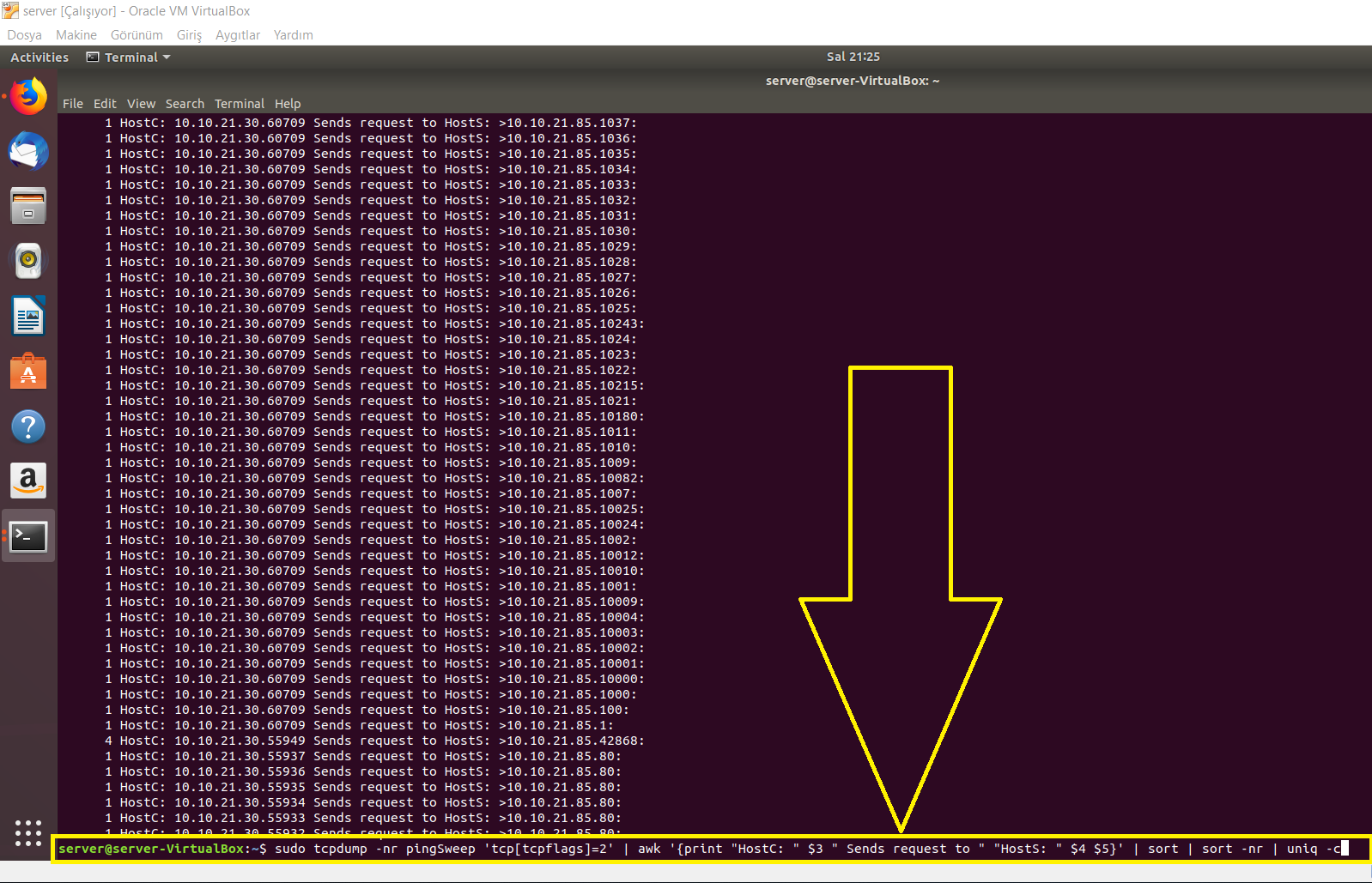
However, ports scan attack drops the connection when the first packet is returned from the server. This is called a "half-open" scan and used to be promoted as a way to surreptitiously scan for ports since the application associated with that port would not receive the traffic, because the connection is never completed.

So to detect port scan we must filter the tcpdump to get SYN and ACK requests.

**sudo Tcpdump pingSweep ‘tcp[tcpflags]=2’ | awk ‘print “HostC: “ $3 “ Sends request to “ “HostS: “ $4 $5}’ | sort | sort -nr | uniq -c**



As you can see attacker with IP(10.10.21.30) HostC sends a request to every possible port of victim with IP(10.10.21.85) HostS. We can see here Yellow circled ports are checked by the attacker from red circled ports.



This picture shows the SYN requests that are made from some IP and its port number, in this case, 10.10.21.30 makes SYN requests from different ports and after receiving ACK it drops the communication. So let's look at some filters use here;

**‘tcp[tcpflags]=2’** ⇒ filters the SYN requests

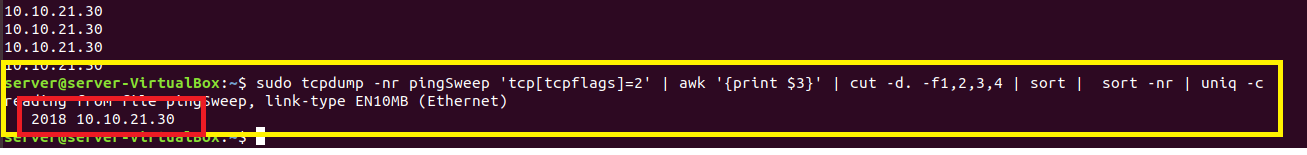
**awk ‘print “HostC: “ $3 “ Sends request to “ “HostS: “ $4 $5}’** ⇒ filters fields with a nice way so we can observe result better.

**sort | sort -nr | uniq -c ⇒** uniq -c cuts unique IPs which in this case is we have only one IP. then the output is sorted.

**pingSweep ⇒** name of the pcap file to read

Another method to find is to use below command which filters the pcap file and outputs only the port scanning IPs.

**$ tcpdump -nr traffic.pcap 'tcp[tcpflags]=2' | awk '{print $3}' | cut -d. -f1,2,3,4 | sort | uniq -c | sort -nr**



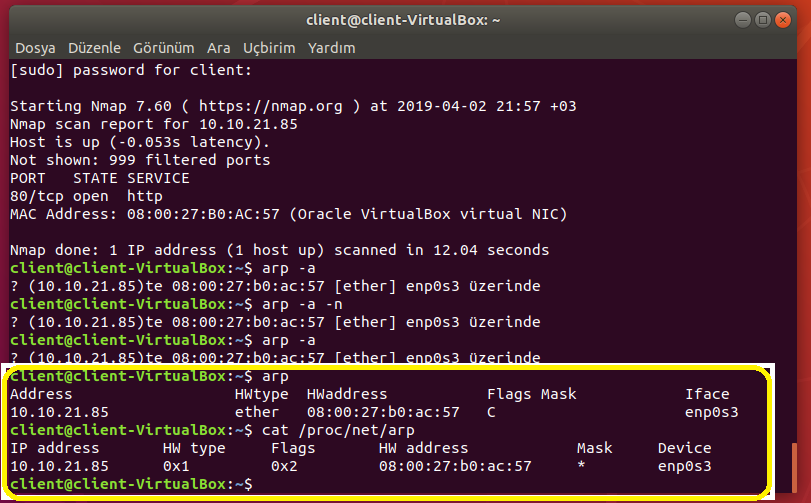
This command cuts the unwanted outputs and only outputs the IP(10.10.21.30) which is the attacker’s IP(HostC).

**T3) Arp Table of HostC**

Two commands used which are:

**arp**

and,

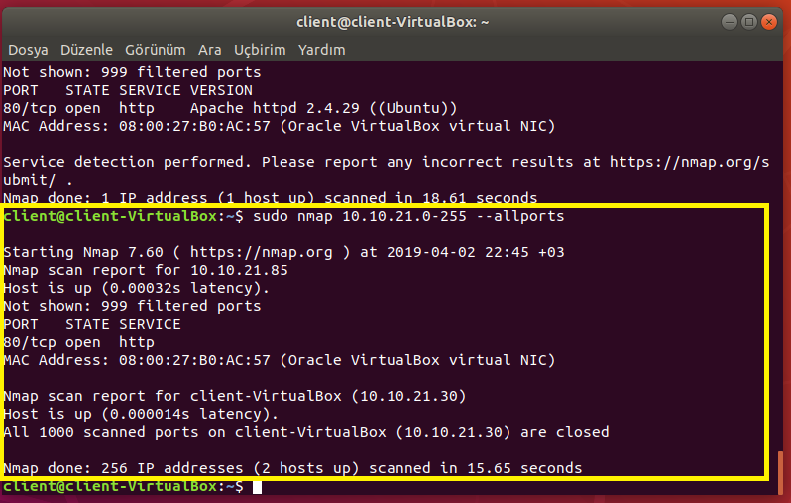
**cat /proc/net/arp**

These two commands are basicly same and they do the same thing. **arp**, displays and modifies entries in the Address Resolution Protocol (ARP) cache, which contains one or more tables that are used to store IP addresses and their resolved Ethernet or Token Ring physical addresses. In this closed network, we have only one entry in our arp table which is the 10.10.21.85 and its HW address(MAC address) is stored. The flags indicate if the mac address has been learned, manually set, published (announced by another node than the requested) or is incomplete.

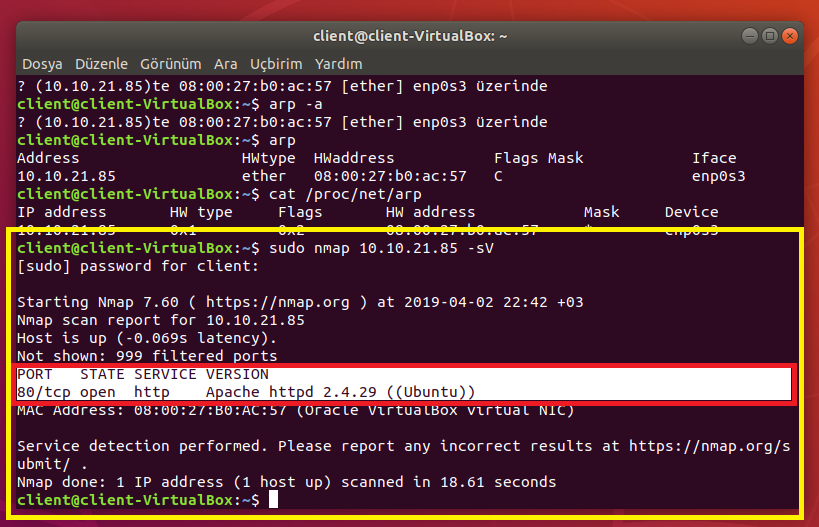
* 0x0 incomplete
* 0x2 complete
* 0x6 complete and manually set

so we can see that MAC address is learned and set by automaticly and it is complete by the flag 0x2. Also, we can learn the interface that is used which enp0s3. And HWtype shows the hardware type which is ethernet card.

**T4) Version of Web Server**

**sudo nmap 10.10.21.0-255 --allports**

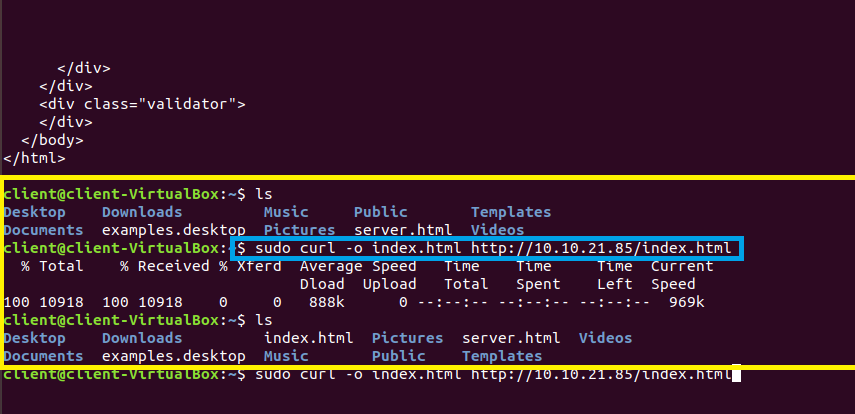
**sudo nmap 10.10.21.85 -sV**



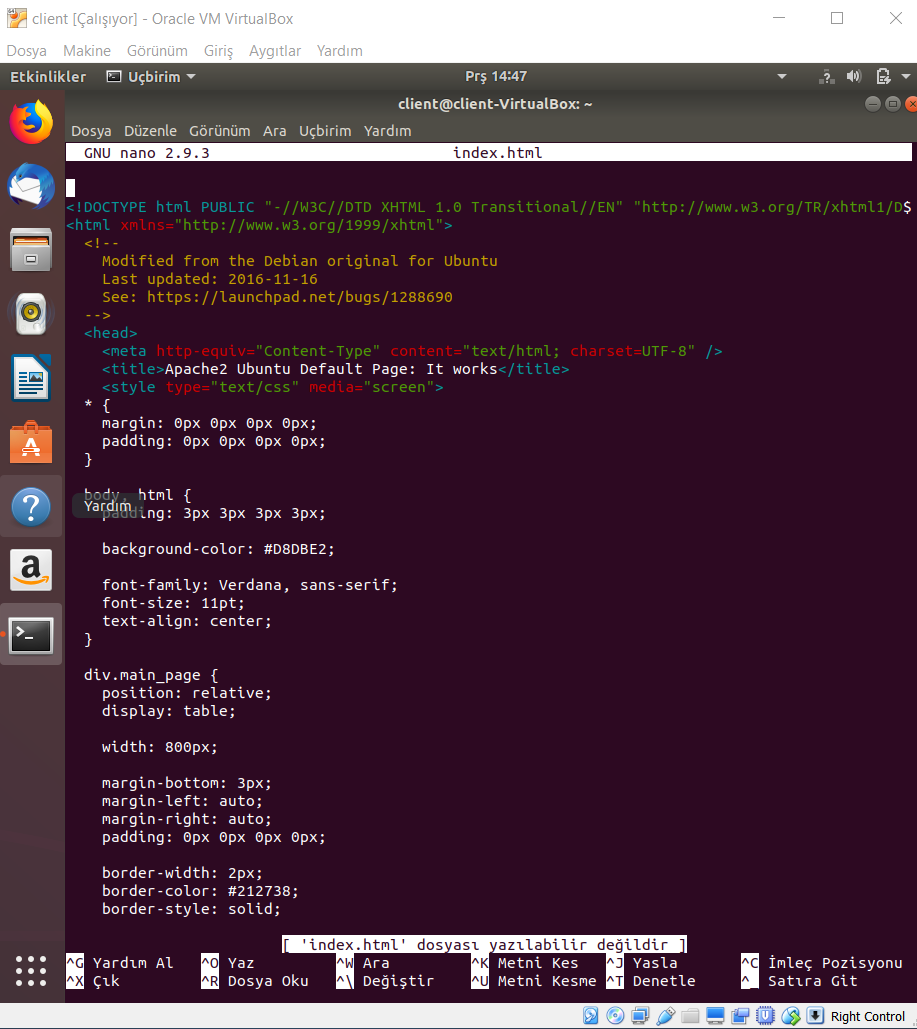
Clients finds that a webserver is open at 10.10.21.85 and start a -sV find command which gives the version of webserver, its state, service and default port. In here we found out that server version is Apache httpd 2.4.29 which works on ubuntu machine.

**T5) Valid page request**

**sudo curl -o index.html** [**http://10.10.21.85/index.html**](http://10.10.21.85/index.html)

****

In this part command written above gets a page request from the webserver, which is index page, and saves it as index.html document. Also download speed, total number of bytes received given as well.

****

This is the content of the index.html that we recorded on our computer. As you can see this is an valid page.

**T6) Pages From Server**

**sudo curl -o login.html** [**http://10.10.21.95/login.html**](http://10.10.21.95.login.html)

**sudo curl -o login.php** [**http://10.10.21.95/login.**](http://10.10.21.95.login.html)**php**

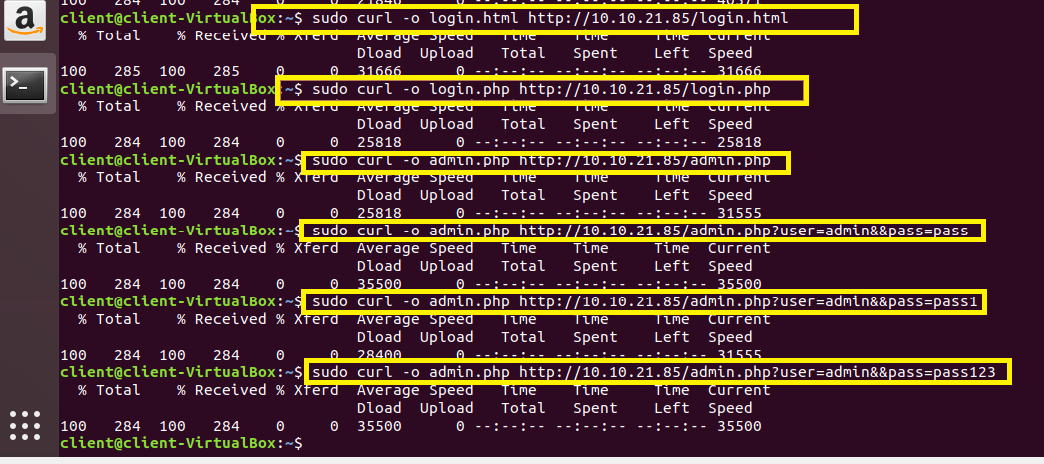
**sudo curl -o admin.html** [**http://10.10.21.95/admin.html**](http://10.10.21.95.login.html)

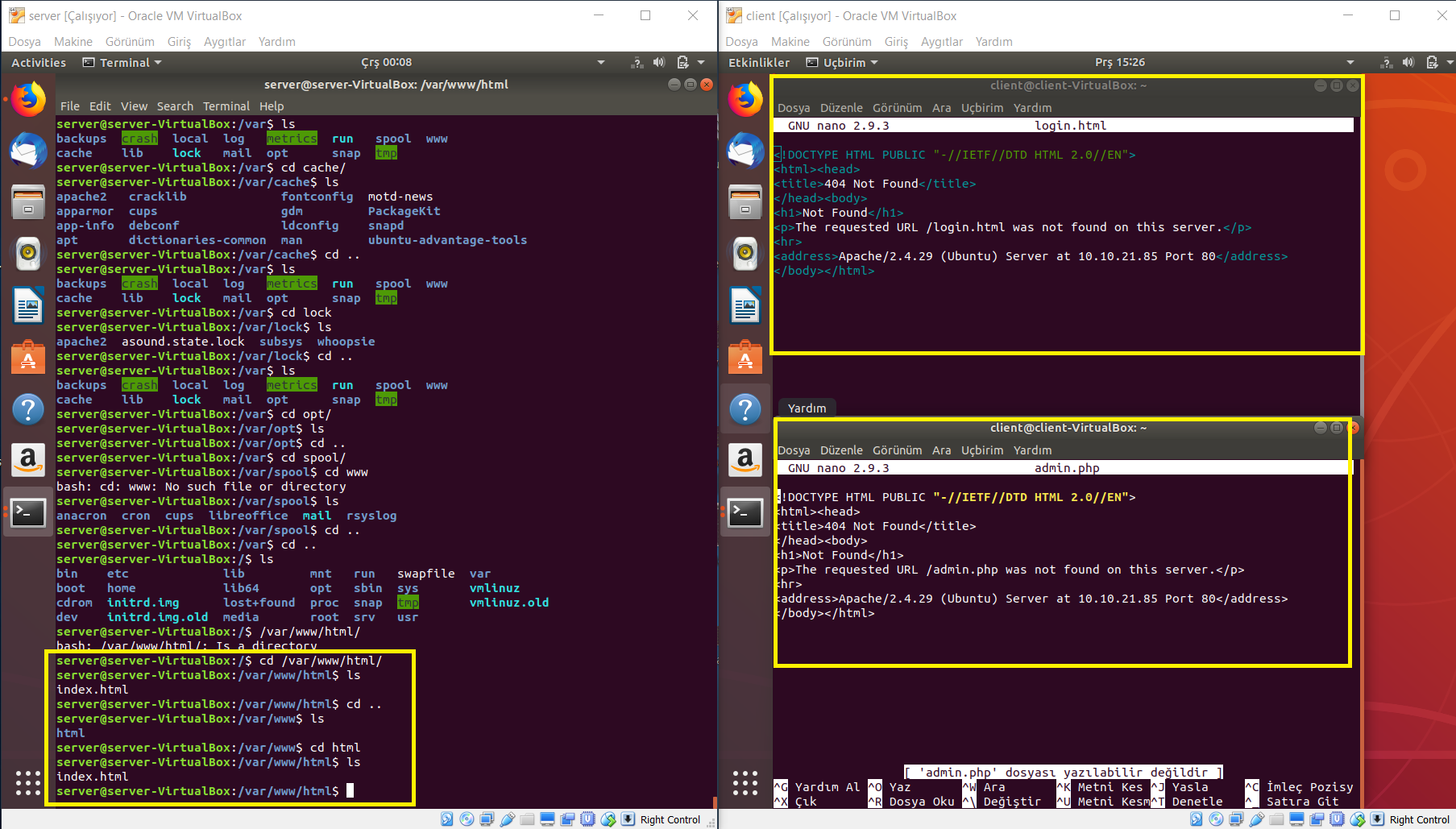
**sudo curl -o admin.php** [**http://10.10.21.95/admin.**](http://10.10.21.95.login.html)**php**

**sudo curl -o admin.php** [**http://10.10.21.95/admin.**](http://10.10.21.95.login.html)**php?user=admin&&pass=pass**

[**http://10.10.21.95/admin.**](http://10.10.21.95.login.html)**php?user=admin&&pass=pass1**

[**http://10.10.21.95/admin.**](http://10.10.21.95.login.html)**php?user=admin&&pass=pass123**

****

**** As above pictures show curl requests are not valid because server does not have those documents that is shown in the server ls command. After pages requested we can see that error message is recorded on the documents instead of the supposed pages because server does not have those pages.